OIPE JC41
OCT 06 2003
PATENT & TRADEMARK OFFICE

2182

# TRANSMITTAL FORM

*(to be used for all correspondence after initial filing)*

| | |
|---|---|
| Application No. | 09/995,285 |
| Filing Date | November 27, 2001 |
| First Named Inventor | Aaron Mar |
| Art Unit | 2182 |
| Examiner Name | |
| Total Number of Pages in This Submission | 35 |
| Attorney Docket Number | 4906P054 |

RECEIVED
OCT 07 2003
Technology Center 2100

## ENCLOSURES *(check all that apply)*

- ☐ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment / Response
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
  - ☐ PTO/SB/08
- ☒ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
  - ☐ Basic Filing Fee
  - ☐ Declaration/POA
  - ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s)
- ☐ After Allowance Communication to Group
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to Group **(Appeal Notice, Brief, Reply Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Other Enclosure(s) *(please identify below):*

Request for Priority

Remarks

## SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

| | |
|---|---|
| Firm or Individual name | Daniel M. DeVos, Reg. No. 37,813<br>BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP |
| Signature | |
| Date | 10/3/03 |

## CERTIFICATE OF MAILING/TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

| | | | |
|---|---|---|---|
| Typed or printed name | Jane Wolfe | | |
| Signature | Jane Wolfe | Date | 10-3-03 |

Based on PTO/SB/21 (08-03) as modified by Blakely, Sokoloff, Taylor & Zafman (wlr) 09/11/2003.
SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450

OIPE JC41
OCT 06 2003
PATENT & TRADEMARK OFFICE

DOCKET NO.: 4906P054

#5

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

AARON MAR, ET AL.

Application No.: 09/995,285

Filed: November 27, 2001

For: **Method and Apparatus for the Enumeration of Sets of Concurrently Scheduled Events**

Art Group: 2182

Examiner:

RECEIVED
OCT 0 7 2003
Technology Center 2100

Commissioner for Patents
P.O, Box 1450
Alexandria, VA 22313-1450

## REQUEST FOR PRIORITY

Applicant respectfully requests a convention priority for the above-captioned application, namely:

| COUNTRY | APPLICATION NUMBER | DATE OF FILING |
|---|---|---|
| Canada | 2327001 | 27 November 2000 |

☒ A certified copy of the document is being submitted herewith.

Respectfully submitted,

Blakely, Sokoloff, Taylor & Zafman LLP

Dated: _____

_____
Daniel M. DeVos, Reg. No. 37,813

I hereby certify that this correspondence is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

*Jane Wolfe* 10-3-03

*Jane Wolfe* *Date*

12400 Wilshire Blvd., 7th Floor
Los Angeles, California 90025
Telephone: (408) 720-8300

Office de la propriété intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An Agency of Industry Canada

RECEIVED
OCT 0 7 2003
Technology Center 2100

# Bureau canadien des brevets
## Certification

# Canadian Patent Office
## Certification

La présente atteste que les documents ci-joints, dont la liste figure ci-dessous, sont des copies authentiques des documents déposés au Bureau des brevets.

This is to certify that the documents attached hereto and identified below are true copies of the documents on file in the Patent Office.

Specification and Drawings, as originally filed, with Application for Patent Serial No: **2,327,001**, on November 27, 2000, by **REDBACK NETWORKS SYSTEMS CANADA INC.**, assignee of Aaron Mar, Ronald Westfall and Gary Robinson, for "Policy Schedule Verification Method and Apparatus".

Sylvie Grégoire
Agent certificateur/Certifying Officer

August 19, 2003
Date

Canada

(CIPO 68)
04-09-02

OPIC CIPO

# METHOD AND APPARATUS FOR THE ENUMERATION OF SETS OF CONCURRENT SCHEDULED EVENTS

## Field of the Invention

This invention relates to automated scheduling systems. In particular, the invention relates to methods and apparatus for enumerating concurrent events in systems of schedules. The invention has application in the field of scheduling generally and has particular application in the field of data communication networks. The invention may be applied in scheduling Quality of Service (QoS) policies in packet handling devices in communications networks.

## Background of the Invention

In many fields of endeavor, it is common to schedule a series of events. An event might be an educational course, a work task, a television program, etc. It is not uncommon for there to be multiple such schedules in effect concurrently. For example there might be a separate schedule for each lecture room in a university, each work party in a construction crew, or each television channel in a viewing area. Schedules can be quite complex given the interruption of regularly scheduled events by special events. In the field of data communications, a QoS policy might be scheduled for activation during specified intervals. Each router or packet forwarding device in a data communications network, or even each interface to a data link within such a device, might have its own set of QoS policies which are activated according to its unique schedule.

There can be dependencies between schedules that prevent an event on one schedule from being scheduled concurrently with an event on another schedule. There is a need for validation methods which can be used to detect such schedule conflicts.

As a preliminary to validation, it is necessary to be able to identify a set of events that are in progress simultaneously. Moreover, for a given group of schedules, it is

necessary to enumerate all such sets of concurrent events from the earliest to the latest date and time in any of the schedules in the group. This invention relates to the efficient enumeration of all sets of concurrent events given a group of schedules.

More specifically the inventors have applied this invention to the validation of policies in a data communication network. Quality of Service ("QoS") policies control the QoS (bandwidth, delay, jitter, etc.) that is enforced by routers and other packet forwarding devices. QoS policies can be scheduled for activation in various parts of the network at various dates and times. Packets passing through the network must experience a consistent set of QoS policies in order to be provided appropriate treatment. This invention is used to enumerate all of the sets of QoS policies that may be concurrently active. Each set can then be validated for consistency.

Schedules can be quite complex. First of all there are at least three types of events:

- a single-shot event that occurs once starting at a specified date and time and ending at another specified date and time (e.g. a television special)
- a recurring event that occurs on some periodic basis between an overall start date and time and an overall end date and time (e.g. a mini-series starting May 1 and running through June 30 that occurs every Monday and Tuesday night between 8:00 and 9:00 PM)
- a recurring event that occurs on some aperiodic basis between an overall start date and time and an overall end date and time (e.g. a meeting that takes place in a lecture room on the 1st and the 15th of each month between 7:00 and 10:00 PM).
- any of the above types of events where the overall end date and time is not known until the event is actually terminated (e.g. a Rotary club meeting that occurs on the third Tuesday of each month until the Rotary club ceases operations).

Secondly, one event can preempt another. For example, regular television network programming may be preempted by coverage of the Academy Awards ceremony. A library in

a school might have an open-ended schedule for its use as a library except when preempted occasionally for the writing of exams.

In a data communication network, there would typically be one default QoS policy that is active with an unspecified termination date and time. This QoS policy might be interrupted by a QoS policy with a recurring schedule that specifies the QoS for an enterprise during its normal weekday working hours. Either of these QoS policies might be interrupted by a single-shot QoS policy that specifies the QoS for a specific event (a one-time videoconference call).

Each schedule is associated with a resource whose use is controlled by the schedule. For example, the resource might be a television channel or a lecture hall. In a data communication network, the resource might be the central packet forwarding processor in a router or packet handling device. The schedule controls when QoS policies are applied to packets transiting through the device. For example, Figure 1 shows several routers 10 - 16 whose central packet forwarding processors are controlled by scheduled QoS policies. Alternatively, each interface to a data link might have sufficient computing power for packet forwarding that the resource is the limited bandwidth of each data link leaving the router or packet handling device. In this case, each interface might have its own schedule of QoS policies. For example, Figure 2 shows a router or packet handling device where the packet flows **121 - 126** being forwarded out of output interfaces **102 - 107** are controlled by QoS policies **111 - 116.** The packets originate at input interface **101**.

Each schedule governs the use of its associated resource. The sequence of television programs that can be watched on a television channel is governed by a schedule. The lectures that can be attended in a lecture hall are governed by a schedule. The QoS policy enforced by the central packet forwarding processor or for each interface in a router or packet handling device is governed by a schedule.

A resource might be controlled by a single master schedule. This schedule might contain all events including multiple levels of preempting events. More realistically a

master schedule is constructed from multiple partial schedules. For example, Figure 3 indicates that a television station might develop its schedule of regular programs for the fall season **200** sometime in the summer. The network, of which the television station is an affiliate, might largely determine this schedule. Overlaid on this schedule will be a specialized schedule **201** for major events such as the Olympics. Schedule **201** preempts schedule **200**. One can imagine that when the Olympics were held in Atlanta, Georgia several years ago, the schedule of Olympic event and news coverage was unique for the Atlanta affiliate. Overlaid on top of these schedules is yet another schedule **202** pulled together at the last moment to handle breaking news events (e.g. the interviews and coverage immediately following an airplane crash).

In a router or packet handling device, a master schedule might be constructed from several simpler schedules. At the lowest level there is a simple schedule of default QoS policies that specify default treatment of packets when no other policies are active. Layered on top of this, for example, is a slowly evolving schedule of QoS policies to handle long duration data services that were subscribed to well in advance. Layered on top of this is a quickly evolving schedule of QoS policies to handle one-time, short duration data services that are subscribed to on a dynamic basis just before the service is used.

Operational efficiencies can be achieved by constructing a master schedule from simpler schedules. For example, the individuals constructing a detailed schedule of coverage of Olympic events probably do not want to worry about the interaction of this detailed schedule with the coarser-grained schedule of regular programs. Instead, periods of time would be blocked off in the regular schedule for Olympic coverage. The individuals planning the detailed Olympic schedule can then construct a more detailed schedule within the allotted periods of time.

When a master schedule is constructed from multiple simpler schedules, each schedule can be assigned a priority. A higher priority schedule interrupts or preempts a lower priority schedule.

In summary, the usage of a set of N resources might be governed by N master schedules, one per resource. Each master schedule might be an amalgamation of multiple simpler

mini-schedules. Each of the mini-schedules has a priority that determines which mini-schedule is currently in effect. Each mini-schedule contains a variety of single-shot and recurring events that may or may not have a termination date and time specified.

There can be dependencies between events that prevent an event on one schedule from being scheduled concurrently with an event on another schedule. Consider multiple work crews, each with their own schedule of work tasks. Certain work tasks might require a certain type of equipment that is in limited supply. Obviously if there are M items of this type of equipment, only M work crews can be performing work tasks concurrently that require this type of equipment.

Similarly consider a university that has multiple lecture schedules, one for each lecture room. It would be obviously inappropriate to schedule two lectures to occur simultaneously where a significant number of students are signed up to attend both lectures. In a router or packet forwarding device, QoS policies contain classification rules which select a subset of the packets passing through the device in order to specify the QoS treatment that those packets are to receive. For example, a QoS policy might specify that all TCP packets destined to TCP port 80 at the IP address 208.216.181.15 should take advantage of a reserved pool of 1 Mbps of bandwidth. It so happens that this policy would match all web browsing traffic (TCP port 80) destined for the www.amazon.com web site (208.216.181.15). Needless to say, another QoS policy in the same router or packet handling device should not specify that this same traffic is ordinary traffic that must compete for whatever bandwidth is available. For example, in Figure 2 QoS policy **112** might contain the above rules specifying that web traffic bound for www.amazon.com should take advantage of 1 Mbps of bandwidth that is reserved for such traffic on output interface **103.** The QoS policy **114** might also contain rules matching the web browsing traffic bound for www.amazon.com, but QoS policy **114** might specify that this traffic should just compete with other traffic for access to the bandwidth of output interface **105.** The router or packet handling device would not know which of the two policies to enforce.

Even if the two mismatching policies are in separate routers or packet handling devices and one device is upstream of the other, there may not be a direct conflict, but the packets will not be consistently provided with the appropriate QoS treatment either.

Validation is used to detect concurrently scheduled events that are incompatible. A validation method is fed a set of concurrent events. The set of events is analyzed to detect events that are incompatible.

For example, the set of lectures that will be held at 2:30 PM on September 20, 2000 might be analyzed to see if there are two or more lectures where the number of double or triple booked students exceeds some threshold value.

As another example, the co-pending commonly owned Canadian patent application entitled POLICY VERIFICATION METHODS AND APPARATUS and filed on 27 November, 2000 describes a policy validation method that can detect conflicts in a set of QoS policies that are scheduled to be active simultaneously. In a complex set of N master schedules, each schedule consisting of prioritized preempting mini-schedules, there will be many different sets of concurrent events that must be validated.

For example, it is not only necessary to validate the set of lectures being held at 2:30 PM, but also those being held at 3:30 PM and 9:00 AM. Validation must not only be performed for each lecture period on September 20, 2000, but also every other day of the school term. A first problem that needs to be solved is to enumerate the sets of concurrent events to be validated.

On the surface this does not seem to be that difficult of a problem. All of the events would be sorted by their scheduled start date and time. Starting at the earliest start date and time and proceeding to the latest end date and time, the sorted events would be analyzed to enumerate the sets of concurrent events.

Unfortunately this method has a serious deficiency. Some events may not have a specified end date and time. Unless the algorithm is designed carefully, there is a chance of ending up in an infinite loop. If all of the events were single-shot events, the

unspecified end dates and times would not be that big a problem. Unfortunately, recurring events introduce a serious problem. Within their overall schedules, recurring events have a periodic or aperiodic pattern of occurrence. If the overall end date and time are not specified, this pattern repeats itself indefinitely.

Consider three trivial schedules. Each schedule has just one recurring event in it. The overall schedule of each recurring event is from January 1, 2000 until the indefinite future. Suppose the pattern of the three recurring events was that shown below:

- Event "A" - from 10:00 AM to 12:00 PM every Monday, Wednesday, and Friday
- Event "B" - from 9:30 AM to 10:30 AM on the 1st and 15th of each month
- Event "C" - from 10:30 AM to 11:30 AM every other day

Which sets of concurrent events should be validated? The combination of the first and third recurring events is fairly easy to figure out. The second event is problematic, since there may be a large number of points in the future where the 1st and 15th of each month falls on a Monday, Wednesday, Friday, or "every other day".

In real life situations there will be a large number of concurrent schedules. For example, a university may have several hundred lecture rooms. In certain data network scenarios, there may be as many as a million routers or packet handling devices, each with several interfaces. Even if each schedule has relatively few events, analysis of the schedules to enumerate the sets of concurrent events may be extremely difficult. A brute force, time-based enumeration of events may easily degenerate into an infinite loop that never terminates. A second problem that needs to be solved is to enumerate the minimum set of unique sets of concurrent events.

Validation of a set of concurrent events may be very computationally expensive. This is certainly true of QoS policy validation. In addition there may be time constraints on how long it takes to perform validation. In the past, it was acceptable to perform such validation in batch mode such that the results might not be available until after several hours of computation. These days, applications are much more interactive. There is a much greater chance that validation has to be performed interactively while a user is waiting for a response to their request to adjust a schedule.

To be efficient a validation process should not waste time validating a set of concurrent events that has been previously validated. If one considers the above three schedules with their simple recurring events, it should be obvious that the same sets of concurrent events are going to repeat themselves over and over.

As a trivially simple example of this, consider two concurrent recurring events:

- from 10:00 AM to 12:00 PM every Monday, Wednesday, and Friday
- from 10:30 AM to 11:30 AM every other day

Unlike the previous three schedules, assume that the overall schedule for these two recurring events is fixed to start on January 1, 2000 and end on December 31, 2000. During the year this schedule is in operation, the first recurring event occurs approximately 52 x 3 = 156 times and the second recurring event occurs approximately 365 / 2 = 182 times. It should be obvious that we do not want to waste time iterating through the year checking the set of events whenever one of these events occurs. That would require validation of approximately 338 sets of events. Keep in mind that there is no commonality between the 156 events and the 182 events, because they start at 10:00 AM and 10:30 AM respectively.

The inventors have determined that this problem can be solved by providing a method that can quickly reach the conclusion that there are just 4 unique sets of concurrent events: neither event in progress, only the first event is in progress, only the second event is in progress, and both events in progress. This results in an 80-fold reduction in validation effort.

The fact that there are only four unique combinations of the two recurring events suggests another simple method for solving the problem. Instead of performing brute force, time-based enumeration of events, one can calculate all of the possible combinations of events and evaluate all of them. Unfortunately there are two problems with this supposedly simple method.

First, the number of possible unique combinations can be quite large. If N resources each have a schedule containing one recurring event, there can be as many as 2N combinations of

these events being in progress or not in progress if the timing of the events is being ignored. It does not take very many resources before it becomes impractical to validate the possible combinations.

As an example of how impractical such an approach can be, consider the following. In a packet handling device developed by the inventors, there can be more than 256 output interfaces that are controlled by QoS policies. Each interface has its own schedule of QoS policies to activate. The above approach would require that more than $2^{256}$ combinations of QoS policies be identified and evaluated. There is not and never will be enough disk storage to even list the possible combinations, much less validate them. In practice, very few of the $2^{256}$ combinations would ever really occur. Most of the interfaces just have a default QoS policy that is scheduled to be always active.

Second, some of the possible unique combinations may be known to conflict. It is only due to the temporal pattern of the various schedules that a conflict is avoided by preventing these combinations from ever occurring. If such a combination were identified in ignorance of the temporal nature of the various schedules and fed to the validation process, it would be incorrectly concluded that there is a conflict in the schedules.

It should be obvious that identifying a minimum set of unique sets of concurrent events to be validated in real life situations is non-trivial. What is needed is a method for enumerating sets of concurrent events from schedules for N resources. The method should be able to accommodate each resource having a master schedule consisting of multiple mini-schedules each with a preemption priority. The method should generate a set having the minimum number of unique sets of concurrent events.

## Brief Description of the Drawings

In drawings used to explain non-limiting embodiments of this invention,

Figure 1 is a schematic network diagram showing a network having a number of devices on which QoS policies may be deployed according to various schedules;

Figure 2 is a block diagram of a single router having a plurality of interfaces each having a QoS policy schedule;

Figure 3 illustrates the preemption of a lowe priority schedule by a higher-priority schedule; and,

Figure 4 is a flow chart illustrating a method according to the invention.

Detailed Description

A currently preferred embodiment of the invention will now be described.

The method of the invention requires the ability to designate points in time when events start and end. In a preferred embodiment of the invention each event has a starting date and time and an ending date and time. In addition to absolute dates and times (e.g. February 7, 2000 8:05 PM), two special values are used:

- Dawn of Time ("DOT") and,
- End of Time ("EOT").

The Dawn of Time represents a date and time that precedes any other specified start or end date and time in any event while the End of Time represents a date and time indefinitely in the future. Events with no specific end date and time will be assigned EOT as their end date and time.

Each single-shot event is represented by a data record containing the following information:

- overall start date and time;
- overall end date and time;
- preemption priority of the associated mini-schedule
- the identity of the resource for which this event is scheduled
- the identity of the single-shot event

The qualifier "overall" has been added to the start and end dates and times of single-shot events to simplify the following discussion by using consistent terminology.

Each periodic recurring event is represented by a data record containing the following information:

- overall start date and time
- overall end date and time
- a date and time pattern for occurrence of the event within the overall period
  - o the length of the pattern's period
  - o the start date and time of the pattern instance that is in progress at the time of the overall start date and time of the event
  - o a list of one or more pairs of start dates and times and end dates and times within the pattern
- preemption priority of the associated mini-schedule
- the identity of the resource for which this event is scheduled
- the identity of the recurring event.

Within the two types of data record, event IDs serve as a reference to more detailed information about the event.

In addition to event data records, a time period data record ("TPDR") is defined to record those events that can occur within the time period covered by the TPDR. The TPDR contains the following information:

- start date and time of the time period
- end date and time of the time period
- a list of the identities of one or more event data records for events that can occur within the time period

An event data record can only be referenced in a TPDR if the event data record's overall start date and time is earlier or the same as the TPDR's start date and time. Similarly, an event data record can only be referenced in a TPDR if the event data record's overall end date and time is later or the same as the TPDR's end date and time. Those individuals skilled in the art will appreciate that other data record formats can be defined without changing the essential nature of the invention.

As shown in Figure 4, the method of the invention starts by collapsing the mini-schedules associated with each of N resources into a master schedule for each resource (sub-method **300**). The method continues to collapse the N resource master schedules into a single universal master schedule (sub-method **400**). The method concludes by iterating through each time period in the universal master schedule to identify sets of concurrent events (sub-method **500**).

The method of the invention starts by collapsing the mini-schedules associated with each of N resources into a master schedule for each resource (sub-method **300** in Figure 4). If a resource has only a single schedule, it is treated as if it is the only mini-schedule for that resource. The collapsing process is performed separately for each resource.
The resulting resource master schedule will consist of a sequence of TPDRs ordered by their start dates and times. The periods covered by the various TPDRs are not allowed to overlap. Each TPDR will reference the event data records of events that can occur within the time period covered by the TPDR. An event data record may be referenced by a sub-sequence of one or more TPDRs.

During the collapse, each event is transformed into an event data record of the appropriate type that is referenced by one or more TPDRs.

For each of N resources, a data structure is created to hold the resource's master schedule. An appropriate data structure must be able to hold an ordered sequence of TPDRs. Ordering is by start date and time of the time period. The data structure should provide O(log N) or better performance for the insertion of TPDRs. On request, the data structure should return with O(log N) performance or better the TPDR whose time period contains a given date and time. The data structure should also support ordered iteration through the TPDRs in the data structure. Examples of an appropriate data structure include a B-tree and a skip list.

For each of N resources, a single TPDR is created whose start date and time is DOT and end date and time is EOT. The TPDR references no event records initially. The TPDR for each resource is inserted into the resource's master schedule data structure. For each of N resources, the mini-schedules corresponding to the resource are sorted by preemption priority. For each of N resources, the sorted mini-schedules are processed in order starting with the lowest priority and ending with the highest priority. The order in which mini-schedules of the same priority are processed does not matter.

The events in each mini-schedule are processed. The order of processing is not important. For each single-shot event, a single-shot event data record is created. For each recurring event, a recurring event data record is created. If the event has no specified overall end date and time, it is assigned EOT as its overall end date and time. It is also allowable for an event to be assigned DOT as its overall start date and time.

The resource master schedule data structure is accessed in order to retrieve the TPDR whose period includes the event's overall start date and time. The event's overall start date and time is compared to the retrieved TPDR's start date and time. The event's overall end date and time is compared to the retrieved TPDR's end date and time.

One of six situations can occur:

1. The event's overall start date and time is identical to the TPDR's start date and time and the event's overall end date and time is earlier than the TPDR's end date and time. In other words, the overall time period of the event corresponds to the leading edge of the TPDR's time period.

In this situation, the TPDR is cloned. The end date and time of the original copy is changed to the overall end date and time of the event. The start date and time of the clone copy is changed to the overall end date and time of the event. The clone copy of the TPDR is inserted into the resource master schedule data structure. The original copy of the TPDR is modified as described below to reference the event data record.

2. The event's overall start date and time is identical to the TPDR's start date and time and the event's overall end date and time is identical to the TPDR's end date and time. In other words, the overall time period of the event corresponds perfectly with the time period of the TPDR.

In this situation, the TPDR is modified as described below to reference the event data record.

3. The event's overall start date and time is identical to the TPDR's start date and time and the event's overall end date and time is later than the TPDR's end date and time. In other words, the time period of the TPDR corresponds to the leading edge of the event's overall time period.

In this situation, the TPDR is modified as described below to reference the event data record.

4. The event's overall start date and time is later than the TPDR's start date and time and the event's overall end date and time is earlier than the TPDR's end date and time. In other words, the overall time period of the event is embedded in the middle of the TPDR's time period.

In this situation, the TPDR is cloned twice. The end date and time of the original copy is changed to the overall start date and time of the event. The start date and time of the first clone copy is changed to the overall start date and time of the event. The end date and time of the first clone copy is changed to the overall end date and time of the event. The start date and time of the second clone copy is changed to the overall end date and time of the event. The two clone copies of the TPDR are inserted into the resource master schedule data structure. The first clone copy of the TPDR is modified as described below to reference the event data record.

5. The event's overall start date and time is later than the TPDR's start date and time and the event's overall end date and time is identical to the TPDR's end date and time. In other words, the overall time period of the event corresponds to the trailing edge of the TPDR's time period.

In this situation, the TPDR is cloned. The end date and time of the original copy is changed to the overall start date and time of the event. The start date and time of the clone copy is changed to the overall start date and time of the event. The clone copy of the TPDR is inserted into the resource master schedule data structure. The clone copy of the TPDR is modified as described below to reference the event data record.

6. The event's overall start date and time is later than the TPDR's start date and time and the event's overall end date and time is later than the TPDR's end date and time. In other words, the overall time period of the event partially overlaps the trailing edge of the TPDR's time period.

In this situation, the TPDR is cloned. The end date and time of the original copy is changed to the overall start date and time of the event. The start date and time of the clone copy is changed to the overall start date and time of the event. The clone copy of the TPDR is inserted into the resource master schedule data structure. The clone copy of the TPDR is modified as described below to reference the event data record.

In all six situations, one of the TPDRs is modified to include a reference to the event data record. If the event is a single-shot event, the reference to its single-shot event data record replaces any other references to event data records in the TPDR. This indicates that the single-shot event is preempting any other lower priority event that was otherwise going to occur during the time period covered by the TPDR. An error is declared if the single-shot event has the same priority as any other events referenced by the TPDR. If the event is a recurring event, the reference to its recurring event data record is added to the list of references to event data records in the TPDR. This indicates that all of the referenced events

can occur in the time period covered by the TPDR. An error is declared if the recurring event has the same priority as any single-shot events referenced by the TPDR.

It should be noted that even though a TPDR references an event, it is possible that the event does not occur in the time period covered by the TPDR. The time period covered by the TPDR may be too short for a recurring event to actually occur given that the recurring event's period may be quite long and the duration that the event is actually in progress may be short. A low priority single-shot event may never get a chance to occur, because there is always a higher priority recurring event preempting it in the time period.

In situations 3 and 6 above, the overall time period of the event extended later than the time period of the TPDR into at least one following TPDR and possibly additional TPDRs. The successively following TPDRs in the resource master schedule data structure are iteratively processed up to and including the TPDR whose time period includes the overall end date and time of the event. In the following description, each of the successive TPDRs that are being processed is referred to as an overlapping TPDR, because its time period overlaps that of the event.

If the event's overall end date and time is later than or equal to the end date and time of the overlapping TPDR, the event completely overlaps the time period covered by the overlapping TPDR. It is only necessary to modify the overlapping TPDR to include a reference to the event data record. If the event is a single-shot event, the reference to its single-shot event data record replaces any other references to event data records in the overlapping TPDR. If the event is a recurring event, the reference to its recurring event data record is added to the list of references to event data records in the overlapping TPDR. Errors are declared for events of equal priority as described above.

If the event's overall end date and time is earlier than the end date and time of the overlapping TPDR, the event partially overlaps the time period covered by the overlapping TPDR. The overlapping TPDR is cloned. The end date and time of the original copy is changed to the overall end date and time of the event. The start date and time of the

clone copy is changed to the overall end date and time of the event. The clone copy of the overlapping TPDR is inserted into the resource master schedule data structure.

The original copy of the overlapping TPDR is modified to include a reference to the event data record. If the event is a single-shot event, the reference to its single-shot event data record replaces any other references to event data records in the original copy of the overlapping TPDR. If the event is a recurring event, the reference to its recurring event data record is added to the list of references to event data records in the original copy of the overlapping TPDR. Errors are declared for events of equal priority as described above.

Processing of events from mini-schedules of increasingly higher priority continues until all of the mini-schedules for a given resource have been processed. The resulting resource master schedule data structure will contain one or more TPDRs, each of which references zero or more event data records. A TPDR will reference zero event data records, if there is no event whose overall time period includes the time period covered by the TPDR. With the possible exceptions of the start date and time of the first TPDR and the end date and time of the last TPDR in the resource master schedule data structure, the start and end dates and times of all TPDRs will coincide with the start and end dates and times of events. The corollary is that for each event's overall start date and time, there will be a TPDR with that date and time as its end date and time followed immediately by another TPDR with that date and time as its start date and time. Similarly, for each event's overall end date and time, there will be a TPDR with that date and time as its end date and time followed immediately by another TPDR with that date and time as its start date and time. For any event referenced by a TPDR, the time period covered by the TPDR will be identical to or a sub-period of the event's overall time period.

Processing of resources continues until a resource master schedule has been constructed for each resource. The method of the invention proceeds by collapsing the N resource master schedules into a single universal master schedule (sub-method **400** in Figure 4).

The first of the N resource master schedule data structures is re-designated as the single universal master schedule data structure.

Each of the remaining N-1 resource master schedule data structures is iteratively collapsed into the universal master schedule data structure. The order in which the remaining N-1 resource master schedule data structures is collapsed into the universal master schedule data structure does not matter.

For a given one of the remaining N-1 resource master schedule data structures, the TPDRs are extracted in order from the resource master schedule data structure. Each extracted TPDR A is collapsed into the universal master schedule data structure as follows. The universal master schedule data structure is accessed in order to retrieve the TPDR B whose time period includes the start date and time of TPDR A. The start date and time of TPDR A is compared to the start date and time of TPDR B. The end date and time of TPDR A is compared to the end date and time of TPDR B.

One of six situations can occur:

7. The start date and time of TPDR A is identical to the start date and time of TPDR B and the end date and time of TPDR A is earlier than the end date and time of TPDR B. In other words, the time period of TPDR A corresponds to the leading edge of the time period of TPDR B.

In this situation, TPDR B is cloned. The end date and time of the original copy of TPDR B is changed to the end date and time of TPDR A. The start date and time of the clone copy of TPDR B is changed to the end date and time of TPDR A. The clone copy of TPDR B is inserted into the universal master schedule data structure. The original copy of TPDR B is modified as described below to include references to the event data records in TPDR A.

8. The start date and time of TPDR A is identical to the start date and time of TPDR B and the end date and time of TPDR A is identical to the end date and time of TPDR B.

In other words, the time period of TPDR A corresponds perfectly with the time period of TPDR B.

In this situation, TPDR B is modified as described below to include references to the event data records in TPDR A.

9. The start date and time of TPDR A is identical to the start date and time of TPDR B and the end date and time of TPDR A is later than the end date and time of TPDR B. In other words, the time period of TPDR B corresponds to the leading edge of the time period of TPDR A.

In this situation, TPDR B is modified as described below to include references to the event data records in TPDR A.

10. The start date and time of TPDR A is later than the start date and time of TPDR B and the end date and time of TPDR A is earlier than the end date and time of TPDR B. In other words, the time period of TPDR A is embedded in the middle of the time period of TPDR B.

In this situation, TPDR B is cloned twice. The end date and time of the original copy of TPDR B is changed to the start date and time of TPDR A. The start date and time of the first clone copy of TPDR B is changed to the start date and time of TPDR A. The end date and time of the first clone copy of TPDR B is changed to the end date and time of TPDR A. The start date and time of the second clone copy of TPDR B is changed to the end date and time of TPDR A. The two clone copies of TPDR B are inserted into the universal master schedule data structure. The first clone copy of TPDR B is modified as described below to include references to the event data records in TPDR A.

11. The start date and time of TPDR A is later than the start date and time of TPDR B and the end date and time of TPDR A is identical to the end date and time of TPDR B. In

other words, the time period of TPDR A corresponds to the trailing edge of the time period of TPDR B.

In this situation, TPDR B is cloned. The end date and time of the original copy of TPDR B is changed to the start date and time of TPDR A. The start date and time of the clone copy of TPDR B is changed to the start date and time of TPDR A. The clone copy of TPDR B is inserted into the universal master schedule data structure. The clone copy of TPDR B is modified as described below to include references to the event data records in TPDR A.

12. The start date and time of TPDR A is later than the start date and time of TPDR B and the end date and time of TPDR A is later than the end date and time of TPDR B. In other words, the time period of TPDR A partially overlaps the trailing edge of the time period of TPDR B.

In this situation, TPDR B is cloned. The end date and time of the original copy of TPDR B is changed to the start date and time of TPDR A. The start date and time of the clone copy of TPDR B is changed to the start date and time of TPDR A. The clone copy of TPDR B is inserted into the universal master schedule data structure. The clone copy of TPDR B is modified as described below to include references to the event data records in TPDR A.

In all six situations, the original or one of the copies of TPDR B is modified to include references to the event data records in TPDR A. The references to event data records in TPDR A are copied and added to the references in the original or copy of TPDR B.

It should be noticed that when references to event data records were being inserted into resource master schedule data structures, the reference to the event data record being inserted might replace any existing references. When TPDR A is being collapsed into the universal master schedule data structure, the event data record references are always added to the existing references in TPDR B. The distinction between the two situations is

that a resource master schedule governs only one resource. A single-shot event in a resource master schedule preempts or replaces any lower priority events. In contrast, the universal master schedule aggregates the events of N resources. The events of each resource occur simultaneously with those of all other resources.

In situations 9 and 12 above, the overall time period of TPDR A extended later than the time period of TPDR B into at least one TPDR following TPDR B and possibly into additional TPDRs. The successively following TPDRs in the universal master schedule data structure are iteratively processed up to and including the TPDR whose time period includes the end date and time of TPDR A. In the following description, each of the successive TPDRs that are being processed is referred to as an overlapping TPDR, because its time period overlaps that of TPDR A.

If the end date and time of TPDR A is later than or equal to the end date and time of the overlapping TPDR, TPDR A completely overlaps the time period covered by the overlapping TPDR. It is only necessary to modify the overlapping TPDR to include references to the event data records in TPDR A. The references to event data records in TPDR A are copied and added to the references in the overlapping TPDR.

If the end date and time of TPDR A is earlier than the end date and time of the overlapping TPDR, TPDR A partially overlaps the time period covered by the overlapping TPDR. The overlapping TPDR is cloned. The end date and time of the original copy of the overlapping TPDR is changed to the end date and time of TPDR A. The start date and time of the clone copy of the overlapping TPDR is changed to the end date and time of TPDR A. The clone copy of the overlapping TPDR is inserted into the universal master schedule data structure.

The original copy of the overlapping TPDR is modified to include references to the event data records in TPDR A. The references to event data records in TPDR A are copied and added to the references in the original copy of the overlapping TPDR.

At this point, TPDR A is discarded.

Extraction of TPDRs from resource master schedule data structures and collapsing of TPDRs into the universal master schedule data structure continues until all of the TPDRs in the remaining N-1 resource master schedule data structures have been processed.

Like the resource master schedule data structures, the resulting universal master schedule data structure will contain one or more TPDRs, each of which references zero or more event data records. The characteristics of the universal master schedule data structure are identical to those of the individual resource master schedule data structures.

The start date and time and end date and time of all TPDRs in the universal master schedule data structure account for all of the overall start date and time and overall end date and time of all events scheduled on all resources. The list of referenced event data records in each TPDR enumerates all of the events that could possibly occur on all of the resources between the start date and time and the end date and time of the time period covered by the TPDR.

Those individuals experienced in the art will appreciate that a variation of the method can be described that directly creates the universal master schedule data structure without performing the preliminary step of creating the N resource master schedule data structures. The alternate method would require a more sophisticated means than a list to track references to event data records in TPDRs.

The method of the invention proceeds to identify all of the unique sets of concurrent events that need to be validated (sub-method 500 in Figure 4). All of the TPDRs in the universal master schedule data structure are processed to identify sets of concurrent events. The order in which TPDRs is processed does not matter.

The set of concurrent events that is in progress at a specified date and time may be represented as a bit vector. The bit vector is made at least large enough to assign one bit to each existing event. The bit corresponding to an event is set to 1 if the event is in

progress at the specified date and time. The bit corresponding to an event is reset to 0 if the event is not in progress at the specified date and time.

In preferred embodiments of the invention, each event is assigned a unique integer event ID whose value numbers upwards from either 0 or 1 depending on the implementation technology being used. Event IDs are preferably reclaimed when events are destroyed and reassigned to new events in order to keep the set of assigned event IDs compact. The value of the event ID is used to determine the bit position within the bit vector that is assigned to the event. The bit vector described above need only be large enough to hold the largest assigned event ID.

In order to generate the minimum number of sets of concurrent events to be validated, a data structure is established to hold an ordered sequence of concurrent event bit vectors. Ordering is by the binary value of the bit vector. The data structure is initially empty. The data structure must provide O(log N) performance for the insertion of bit vectors. If an attempt is made to store a new bit vector into the data structure that is a duplicate of an existing bit vector in the data structure, the new bit vector is not stored in the data structure. Examples of an appropriate data structure would include a B-tree or skip list. This data structure shall hereafter be referred to as the bit vector repository.

As each TPDR in the universal master schedule data structure is processed, its list of event data record references is examined to identify sets of concurrent events. If a TPDR's list of event data record references only includes references to single-shot events then the only set of concurrent events for that TPDR is the set of events referenced by the event data record references. A concurrent event bit vector is created having bits set to 1 that correspond to the listed events. The bit vector is added to the bit vector repository.

If a TPDR's list of event data record references includes any references to recurring events then there may be multiple sets of concurrent events for that TPDR. The sets of concurrent events will depend on which recurring events (and lower priority single-shot events) actually occur during the period of time covered by the TPDR.

This is a fairly controlled situation where brute force, time-based iteration can be used to identify all of the sets of concurrent events. It is still necessary to handle the last TPDR in the universal master schedule data structure that will have an end date and time of EOT (i.e. unspecified). In addition, any of the TPDRs in addition to the last TPDR can cover a very long period of time, so we want to efficiently identify all of the sets of concurrent events.

In order to prevent an infinite loop of time-based iteration, an artificial date and time is established to limit the iteration. All of the recurring events recur on some periodic basis. Different recurring events may have different time periods. A time period is calculated that is the lesser of the time period covered by the TPDR or the least common multiple of all the periods of recurring events referenced by the TPDR. The resulting time period is added to the start date and time of the TPDR to derive the stop date and time for the time-based iteration.

For example, if a TPDR referenced three recurring events with periods of 1.5, 2, and 3 weeks, the least common multiple of their periods would be 6 weeks. If the time period of the TPDR referencing these events were 5 weeks, a 5 week period would be used instead of the 6 week period.

A reference date and time is established whose value is initially the start date and time of the TPDR. An initial set of concurrent events is determined by examining the event data records referenced by the TPDR to determine which events are in progress on the N resources at the start date and time of the TPDR. A bit vector is created representing the set of concurrent events that are in progress at the start date and time of the TPDR. The bit vector is added to the bit vector repository.

A current bit vector reference is established which points to the bit vector representing the set of concurrent events that are in progress at the reference date and time. The current bit vector reference is initially set to point to the bit vector representing the set of concurrent events that are in progress at the start date and time of the TPDR.

Time-based iteration iteratively proceeds to identify additional sets of concurrent events. At the start of each iteration, the recurring event data records referenced by the TPDR are examined to identify the nearest date and time following the reference date and time where an event occurrence either starts or stops. The easiest way to do this is to identify the next significant date and time following the reference date and time for each event referenced by the TPDR. The resulting dates and times are then sorted to find the closest date and time following the reference date and time.

The event data records, for which the selected date and time is significant, are examined. There may be multiple event data records, because multiple events may be scheduled to stop or start at the same time. The event data records for events that are scheduled to stop at the selected date and time shall hereafter be referred to as the current event data records and their events as the current events. The event data records for events that are scheduled to start at the selected date and time shall hereafter be referred to as the next event data records and their events as the next events.

Each current event that is scheduled to stop may or may not correspond to the event that has been in progress for the corresponding resource. The current event may be a lower priority event that has been previously preempted by a higher priority event. The current event may be scheduled to stop while the preempting event is in progress. Similarly, each next event that is scheduled to start may or may not actually be starting for the corresponding resource. The next event may be a lower priority event that has been previously preempted by a higher priority event.

If a current event is the highest priority event for its resource, a transition may be underway to an event that is not a next event, because the event does not have a significant date and time corresponding to the selected date and time. A transition may be underway to an event that was previously preempted.

Similarly, if a next event is the highest priority event for its resource, a transition may be underway from an event that is not a current event. The next event may be preempting the in progress event.

The current and next data records and the other event data records associated with the same resources as the current and next data records are examined to identify which in progress events are stopping and which other events are starting to be in progress. The bit vector referenced by the current bit vector reference is cloned. The bits in the cloned copy of the bit vector, corresponding to in progress events that are stopping, are reset to 0. The bits in the cloned copy of the bit vector, corresponding to events that are starting to be in progress, are set to 1. The modified cloned copy of the bit vector is inserted into the bit vector repository.

The current bit vector reference is set to point to the modified cloned copy of the bit vector. The reference date and time is advanced to the selected date and time. At this point an iteration of the time-based iteration is complete. Further iterations are performed for the same TPDR until the reference date and time points to a date and time that is equal to or greater than the iteration stop date and time calculated earlier.

Upon completion of time-based iteration, processing of the TPDR has been completed. Processing of additional TPDRs in the universal master schedule data structure continues until all TPDRs have been processed.

Once all of the TPDRs in the universal master schedule data structure have been processed, the method of the invention is complete. The bit vector repository holds a minimum unique set of bit vectors representing sets of concurrent events that must be validated.

Preferred implementations of the invention may include a computer system programmed to execute a method of the invention. The invention may also be provided in the form of a program product. The program product may comprise any medium

which carries a set of computer-readable signals corresponding to instructions which, when run on a computer, cause the computer to execute a method of the invention. The program product may be distributed in any of a wide variety of forms. The program product may comprise, for example, physical media such as floppy diskettes, CD ROMs, DVDs, hard disk drives, flash RAM or the like or transmission-type media such as digital or analog communication links. Specific embodiments of the invention may comprise a computer connected to receive QoS schedules from a plurality of packet handling devices, the computer programmed to perform the methods of the invention to identify conflicting events in the QoS policies.

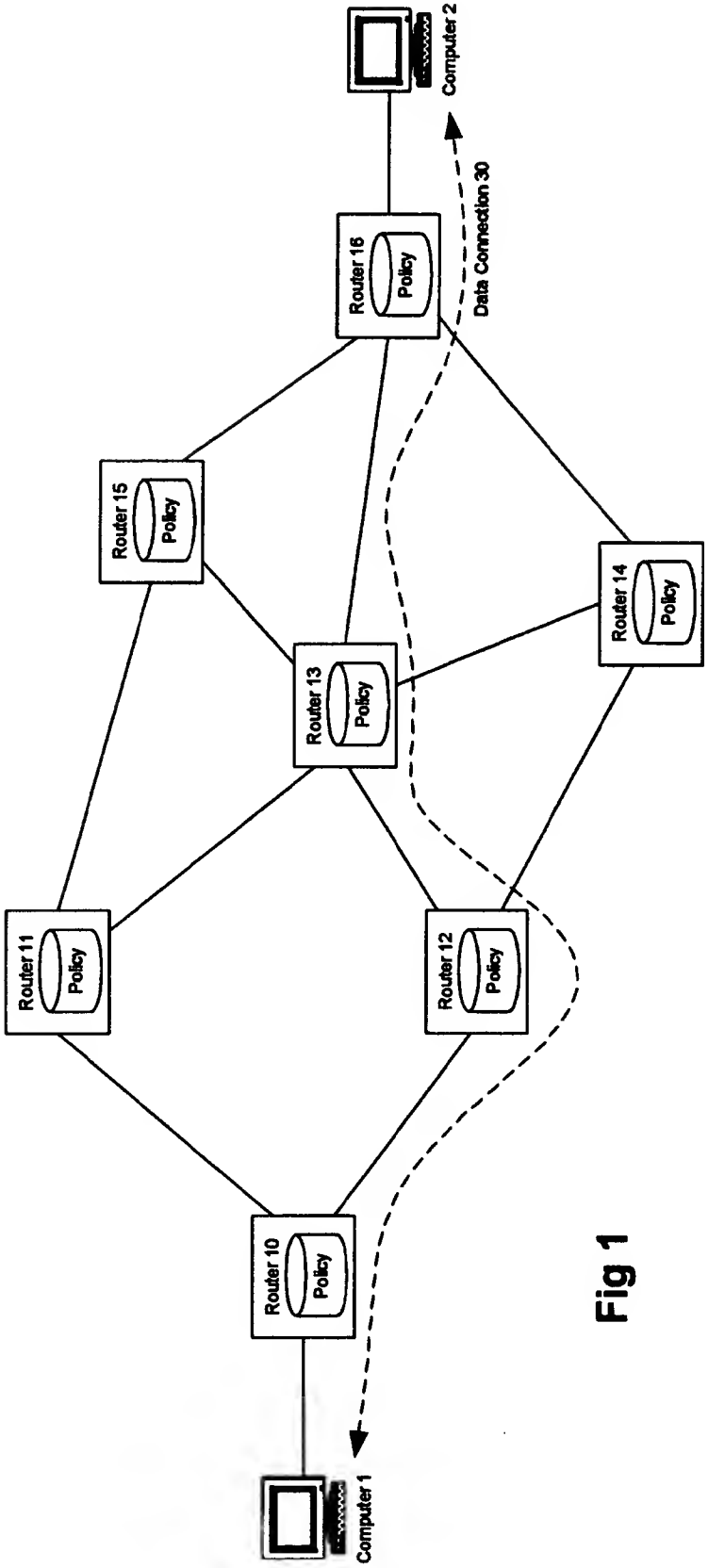
Computer 2
Data Connection 30
Router 16
Policy
Router 15
Policy
Router 14
Policy
Router 13
Policy
Router 11
Policy
Router 12
Policy
Router 10
Policy
Computer 1

Fig 1

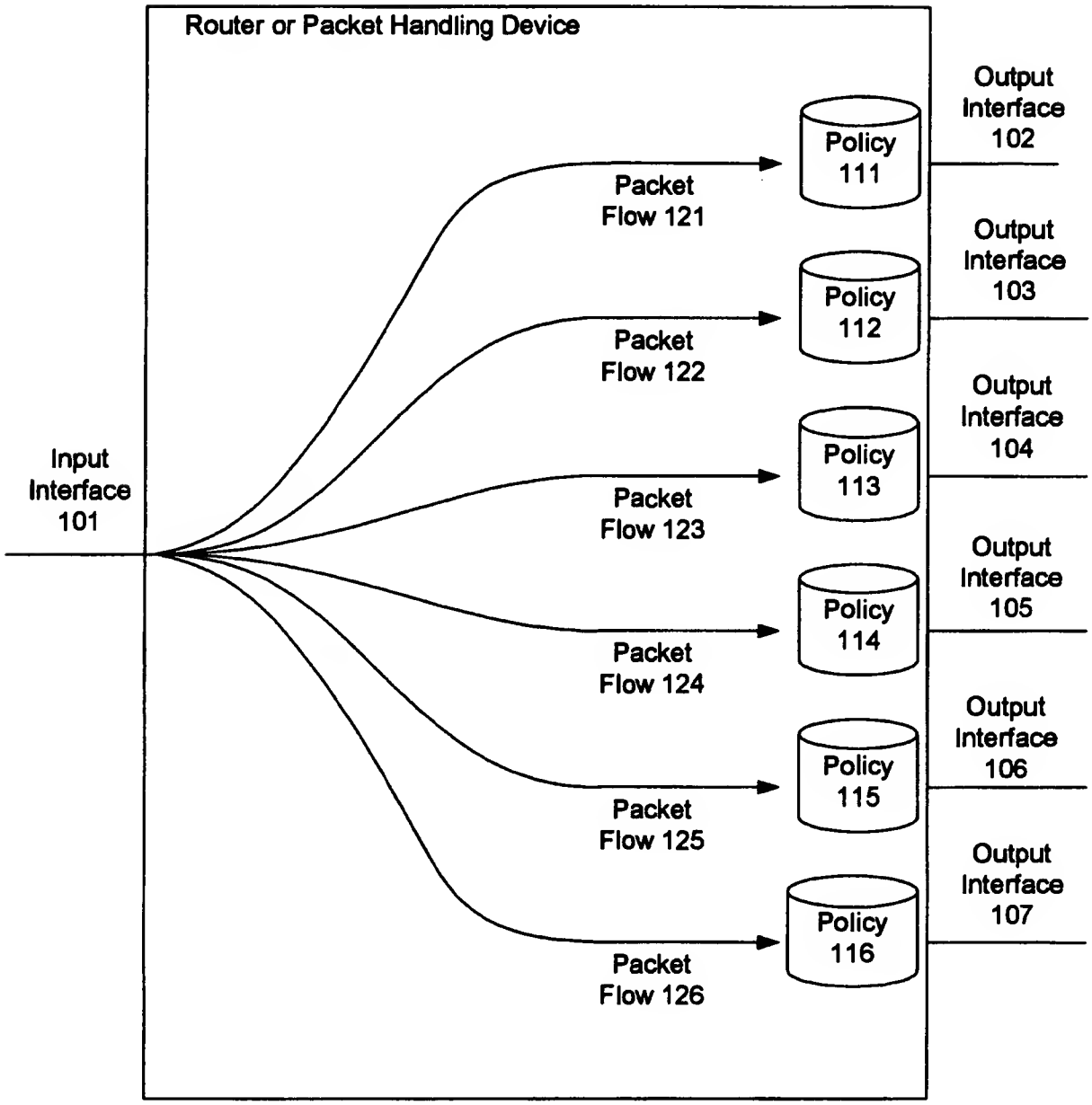
Router or Packet Handling Device
Input Interface 101
Policy 111
Policy 112
Policy 113
Policy 114
Policy 115
Policy 116
Packet Flow 121
Packet Flow 122
Packet Flow 123
Packet Flow 124
Packet Flow 125
Packet Flow 126
Output Interface 102
Output Interface 103
Output Interface 104
Output Interface 105
Output Interface 106
Output Interface 107

Fig 2

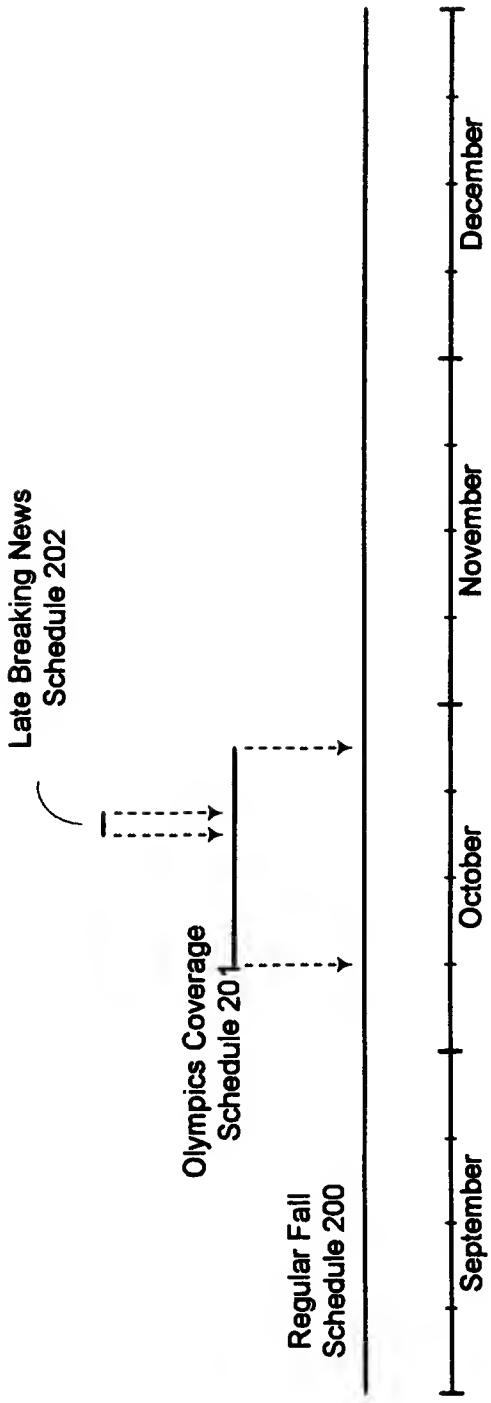
Late Breaking News
Schedule 202
Olympics Coverage
Schedule 201
Regular Fall
Schedule 200
September
October
November
December

Fig 3

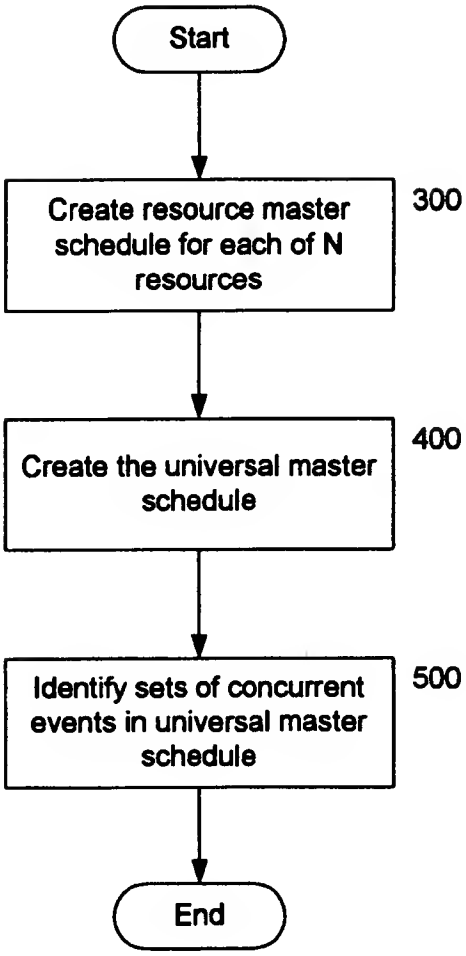
Start
Create resource master schedule for each of N resources
300
Create the universal master schedule
400
Identify sets of concurrent events in universal master schedule
500
End

Fig 4